# Reducing Data Decryption Cost by Broadcast Encryption and Account Assignment for Web Applications

AUTHOR(S):

Kawamoto, Junpei; Ma, Qiang; Yoshikawa, Masatoshi

# Reducing Data Decryption Cost by Broadcast Encryption and Account Assignment for Web Applications

Junpei Kawamoto
j.kawamoto@db.soc.i.kyoto-u.ac.jp

Qiang Ma
qiang@i.kyoto-u.ac.jp

Masatoshi Yoshikawa
yoshikawa@i.kyoto-u.ac.jp
Graduate School of Informatics, Kyoto University
Yoshida-Honmachi, Sakyo-ku, Kyoto, JAPAN

## Abstract

*Protection of user privacy is an important issue of Web applications. Data encryption presents a possible resolution for improving the security level of Web applications. In this paper, we propose a novel mechanism using broadcast encryption and account assignment methods to reduce the decryption cost of Web applications. A notable feature of our mechanism is that the additional function of the application server is not necessary. Moreover, it is easy to apply this method to existing servers to improve their security level. We also show some experimental results to demonstrate the feasibility of our methods.*

## 1. Introduction

Web applications have become notable platforms for the innovative service and Consumer Generated Media (CGM) because they facilitate data sharing and collaboration. However, currently, most Web applications allow the application server (provider) to access user data and use such data for other purposes. In [4] Davida *et al.* have pointed this: *These problems are because the DBMS knows what data it is manipulating.* Therefor it is necessary to preserve the encrypted user data in the server and avoid data inspection with the server, for solving the privacy problem and enabling the user to use the Web application easily and safety. Not only that, it is neccessary to keep the access authority information confidential because access analysis makes servers extrapolate the importance of each data and the priority of users.

As techniques to accomplish those objectives, two directions have been studied. One is the study of methods of retrieving encryption data preserved in server; the other is the study of delivery of the encryption key and data. Among methods of retrieving encryption data preserved in a server [7, 8], an encrypted relational database, that allows retrieval of encrypted data, has been achieved for each tuple by producing a secure index. Among methods of delivery of the encryption key and data [3, 6, 9], the number of encryption keys that each user have to manage is reduced by publishing information of data-sharing group and making the user hierarchy. In this paper, we consider to share the encrypted data and discuss the method of reducing decryption cost without any regard for user hierarchy because in Web application users most always are equivalent.

The naive method of bringing in access authority for encrypted data is described as follows. Data entry $e$ is encrypted with key $k_e$ of the common key cryptosystem. The encrypted key chain, which is the individually encrypted $k_e$ used to encrypt the data entry $e$ with the public key of users who are permitted to have access is added to the encryption data. Only user allowd to access the data can decrypt and get $k_e$ therefore can get the $e$. However, this method requires many keys proportional to the number of the authorize users. Until trying decryption, user cannot know which data in the key chain is able to be decrypted or no data is able to be decrypted because of he/she does not have authority. In short, the long key chain requires the necessary of decrypting many data.

For reducing key chain, a key is allocated at each combination of the user groups. For exsample, let us consider the case of four users, the number of the possible combination group is fifteen. We make in advance fifteen kinds of different keys and assign each key to each group. Data entry $e$ is encrypted with the key assigned for that group, therefore the key chain length is always one. However, in this way, the number of keys increases explosively in proportion to the number of users. Damiani *et al.* introduced the way of reducing the number of keys in [3]. They do not create all possible keys but actually used keys by registering

the static data-sharing group firstly. They also extended to a dynamic condition in [6]. These methods require to publish the authorized users for each data, but this constraint is not enough for protecting user privacy.

The weak points of these conventional methods can be summarized as follows.

- The decryption cost will increase when the encrypted key chain containing access authority information grows.
- The presence of the access authority will not be certified until the encrypted key chain is decrypted. Therefore, it is also necessary to try decryption of data for which one has no access authority.

In this paper, as one solution, we propose a method to reduce the cost of decrypting key chain by using pairing-based broadcast encryption and method to reduce the number of trying to decrypte the data which is not allowed to access by account assignment. The proposed methods do not require special functions of application server. Therefore, it is readily applicable to conventional Web application servers. The experimental results validates the feasibility of our methods.

## 2. Overview and Basic notions

In this section, at first, we introduce the basic notions used in this paper. Then we give the overview of our mechanism for reducing the encryption cost of Web applications.

### 2.1. Data Decryption Cost

It is necessary to decrypt the encrypted key chain containing the access authority information of all data entries to examine the presence of the access authority, when the user want to acquire some data entries. The best case, in which we just need one time for key decryption, is that the key containing the right user access authority is the first item of the encrypted key chain. On the other hand, in the worst case that we do not have access authority of any item, although we decrypted all of the key chain, we cannot obtain any key and any data entry. Generally, the decryption cost $cost(u)$ of user $u$ can be defined as follows.

$$cost(u) = \bar{L} \times N(u) \tag{1}$$

where, $\bar{L}$ is the average size of encrypted key chain. $N(u)$ denotes the number of times user should try decrypting to examine the presence of the authority. We propose the method of reducing both $\bar{L}$ and $N(u)$ to reduce the decryption cost.

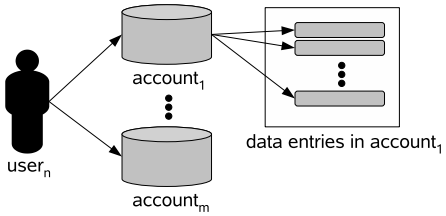



**Figure 1. Users and accounts**

### 2.2. User Account and Account Reference

In a Web application, the notion of *user account* is used to manage data clusters. A user account is authenticated using ID and password. The data clusters used in Web application are generated based on the user account. In this paper, we use the notion of user account for access authority management of data, rather than user identification of the conventional Web applications. A user agent will manage the ID and password of each user account. The user account is not assigned to end user but data cluster. Consequently, users cannot access their accounts without the agent and cannot perform operations that not allowed to that account by the system. The user agent also provide user identification mechanism instead of the mechanism using user account.

Account reference information signifies the correspondence relationship between user accounts and a user. It denotes which user accounts the user has; user accounts are expected to include data items which are accessible by that user. The account reference information is prepared in advance. For instance, when $account_1$ and $account_2$ are in the information of the reference to the account of $user_1$, data to which $user_1$ has access authority are preserved only in these two accounts. Hence, it is not necessary to go to refer other accounts. As a result, the number of data entries that a user is expected to examine for the presence of the access authority can be decreased. In addition, in our mechanism, any user can read and write the account reference information of other users.

The relation between users ant accounts is illustrated in figure 1. Each user references some accounts and each accounts stores some data entries.

### 2.3. Architecture and Dataflow

Figure 2 illustrates the architecture of the Web application server based on our mechanism. When a user tries to add a data entry, the user agent in figure 2 will works as follows.

1. The data entry and its access information (the users who are allowed to access this data entry) will be

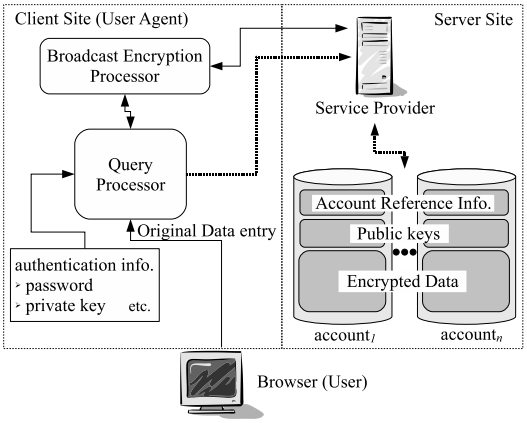



**Figure 2. Architecture and Dataflow**

passed to the Query Processor, which is working on the client site.

2. Query Processor passes Broadcast Encryption Processor information about the users who have been given the access authority.

3. Broadcast Encryption Processor acquires these user's public keys from Service Provider and generates access authority information for each user.

4. Query Processor acquires account reference information from Service Provider. Then it decides the account that should preserve the added data entry using the method described in chapter 3.2.2.

5. Query Processor adds secure indices described in [7] *etc.* and finally sends these data to Service Provider.

Currently, Query Processor processes the query on the encrypted database based using that secure indices. Using the private key, it examine the access authority for each data entries among the query results. Only data entries that have the access authority will be returned to the client or end user.

# 3. Broadcast Encryption and Account Assignment

This section introduces two methods: account assignment for reducing the times of decryption, and pairing-based broadcast encryption for fixing the key length.

## 3.1. Fixing key-length

One problem of the conventional methods is that the encrypted key chain containing access authority information grows to such a great degree in some cases. It makes the decryption inefficient, because, to examine the presence of the authority, it becomes necessary in the worst case to try decryption to all key information. Therefore, we apply pairing-based broadcast encryption [2] to fix the key length. As a result, decryption to examine the presence of the authority needs to be done only once.

The *pairing* function $f$ is defined on a elliptic function and it has two input and one output arguments. Recently, the use of pairing for public-key cryptography is actively studied [5]. Pairing-based Broadcast Encryption [2] is a method that enables a key chain, as an access authority, to be bound by one key independently of the number of authorized users. Security of this pairing-based broadcast encryption boils down to the question of $l-$BDHE assumption. Assuming the hardness of $l-$BDHE assumption, our method has high security and can be proven [2].

The broadcasting encryption approach requires to specify the number of users for creating public/private key pairs. Therefore, given the possibility of adding a user in the future, the following enhancement is necessary.

- Set the number of users to enough large at initialization.
- Create and reserve a lot of public and private key pairs for further usage. When a user has been added, assign the key pair from the reserved ones.

When there is no more reserved key pair, we create multiple access authority information by using broadcast encryption. By this method, the key pair of broadcast encryption increases and, eventually, access authority information will increase if the number of users increases. However, generally, if $n$ is enough large then the frequency of creating a new key pair is less than the increase of users.

## 3.2. Reducing Decryption Candidate Data

The encrypted key chain containing access authority information can be replaced with a single key using broadcast encryption. Therefore, a decryption frequency necessary to examine the presence of the authority is one time always. However, we need to try decryption all the candidate data (encrypted key chains) to confirm whether we have access authority. That is to say, the $N(u)$ is still large and the $cost(u)$ is high (Function (1)). Hence, we propose a user assignment method to reduce the encryption candidates. The basic idea is that, a user is assigned to the user account which containing fewer data entries s/he does not have access authority.

### 3.2.1 Available Information for Account Assignment

It is important that sharing data among the users only and server should not be allowed to access the users' data. If user information on certain data are added as a plaintext, it

is understood which users are sharing data For that reason, the relation of the user will be discovered. Consequently, information on the data sharing cannot be used for account assignment. However, a user who adds data knows about the user to whom he gives the access authority. Therefore, the account assignment is done according at the time when data are added. Moreover, any user can access the account reference information, as described in Chapter 2.2. This account reference information is useful for account allocation.

#### 3.2.2 Assignment Algorithm

The bias of the data included in a user account from account reference information is presumed as follows. Therefore, it is inferred that much data has already been preserved in the account when $n$ is large. Furthermore, when users who are referring to a certain account are few, it is inferred that the amount of data preserved in the account is little. Put simply, it is assumed that the amount of data in the account is associated with the number of account references. It is also assumed that if an account to which a certain user refers increases, then the user has no access authority to any of the data included in the account at that time.

Therefore, a cost is generated when the product of the user who refers newly to the number of data included in the account and the account adds data to the account. For some account $a$, let $n_a^+$ be the number of users who have already referred to account $a$ and let $n_a^{S-}$ be the number of the users in $S$ who have not referred to account $a$. The cost of account $a$ in the set of users $S$ is defined as follows.

$$Cost_S(a) = (n_a^+)^k \cdot n_a^{S-} \ (k : \text{constant}) \qquad (2)$$

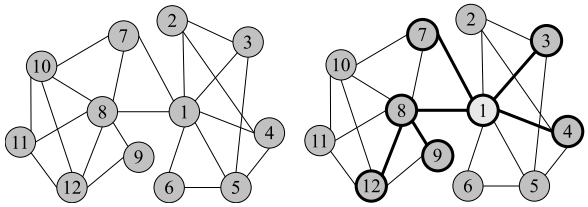
When the account that adds data is selected, data are allocated to the account with the smallest cost.

## 4. Experimental Evaluation

We performed an evaluation experiment by the simulation to confirm the effectiveness of the technique using broadcast encryption and the account assignment proposed in this paper. In this section, we first explain the model of the user relation used in simulation, and then describe the experiment method and the result.

### 4.1. Friendship and Data-sharing Model

In our simulation, the user friendship relation is set as a scale-free friendship graph; data-sharing groups are determined based on this relation. In the graph representing this friendship relation, a node and an edge denote a user and an friendship, respectively. For example, $user_1$ in Figure 3(a) has friendship with $user_2$, $user_3, \cdots, user_8$. That friendship graph is made on the BA-model [1], and data-sharing groups are made by the following process using that graph:




(a) Friendship Graph (b) Data-sharing Group

**Figure 3. Friendship and Data-sharing Model**

1. Choose a node which has not been chosen in a friendship graph $G_n$.
2. Judge whether the selected node will be the head of a new group or not with a probability that is biased so that it is proportional to the number of links.
3. Choose a node connected to some node in the group.
4. Judge whether the chosen node will belong to the group with a probability that is biased so that it is inversely proportional to the number of links.

Because of the assumption that the more friends a user has, the more opportunity for a head there is, choosing a head in step 2) is based on a probability that is biased: it is directly related to the number of links. That is also because of the assumption that the more friends a user has, the greater the probability of belonging to some group already. The judgment whether to be belong to the group in step 4) is based on a probability that is biased so that it is inversely related to the number of links.

Figure 3(b) portrays a case in which the $user_1$ was chosen at step 1) and became a leader of a new group. In addition, at step 3) $user_2, user_3, \cdots, user_8$ are chosen, and $user_3$, $user_4$, $user_7$ and $user_8$ belong to one group. Moreover, $user_9, user_{10}, \cdots, user_{12}$, which were connected to $user_8$ added to group as a head, are chosen at the step 3) and $user_9$ and $user_{12}$ belong to one group.

### 4.2. Simulation and Result

For the simulation, data-sharing groups are created by the user model explained above including $100$, $1,000$ and $10,000$ users. Respectively the procedure of creating groups is executed for three times in each case because of the need to produce many more groups. Each group shares data of the number that is proportional to the number of members. As a result, we calculated the length of the encrypted key chain and the precision of the access authority. We also calculated the distribution of the account reference

| No. | # of groups | Naive method key chain avg. | Naive method precision | Our method precision |
|---|---|---|---|---|
| 100 users | | | | |
| 1 | 112 | 19.0328 | 0.190328 | 0.981911 |
| 2 | 107 | 16.8371 | 0.168371 | 0.991164 |
| 3 | 55 | 13.6349 | 0.136349 | 0.790000 |
| 1,000 users | | | | |
| 1 | 1034 | 27.2675 | 0.0272675 | 0.987987 |
| 2 | 1083 | 30.9286 | 0.0309286 | 0.984673 |
| 3 | 1056 | 31.0023 | 0.0310023 | 0.984935 |
| 10,000 users | | | | |
| 1 | 10563 | 42.4664 | 0.00424664 | 0.988050 |
| 2 | 10432 | 44.4462 | 0.00444462 | 0.989989 |
| 3 | 10642 | 39.9114 | 0.00399114 | 0.988845 |

**Table 1. Result of simulation**

and the precision of the access authority. Here, the precision of the access authority is the rate of the number of the data checked for permission and the number of permitted data. Let $Check(user_i)$ be the number of the data $user_i$ must check to gain permission and $Auth(user_i)$ be the number of data $user_i$ has authority to; the precision of the access authority of $user_i$ $r(user_i)$ is defined as follows.

$$r(user_i) = \frac{Auth(user_i)}{Check(user_i)} \tag{3}$$

The results of this simulation are presented in Table 1. The authority precisions were calculated for the worst case in which each user checks all data the user can get. According to Table 1, it is obvious that our approach can check for permission with only one decryption in any case and the precision of our approach is high. Therefore, in our approach, useless decryptions are executed in few cases.

In figure 4, the horizontal axis shows the number of account references and the vertical axis shows the precision of the access authority. In that figure, the users are also depicted in the corresponding place as a point. Many account references engender a high cost of connection. According to the result, the number of account references in our approach remain at lower values, which is the preferred outcome.

## 5. Data protection on Web application

When apply our methods to Web applications, two important issues will pertain: 1) Who creates the public/private key pair for broadcast encryption? 2) How are the keys delivered? We will discuss these two issues in this section. Hereafter, $n$ denotes the total number of the users; the users are many because of Chapter 3.1. In addition, each $user_i$ has a key pair $(pub_i, private_i)$ of a usual public key cryptosystem.

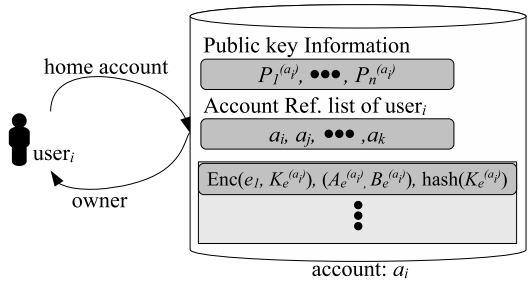



**Figure 5. Setup of the account**

### 5.1. User Account Setup

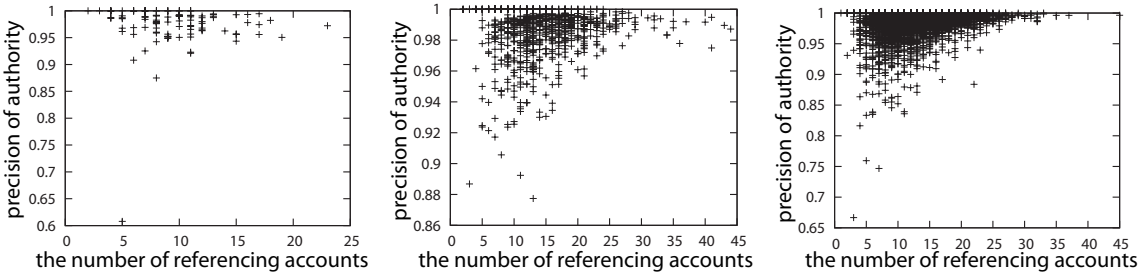
First, the user account is prepared in the number of users $n$; it is assumed to be of $a_1, a_2, \cdots, a_n$. Only account $a_i$ is made to be referred for $user_i$ at first; $a_i$ is called the home account of $user_i$ and $user_i$ is called the owner of account $a_i$. Moreover, data for which the account owner has no access authority are assumed not to be preserved in each account. Each user should have account reference information on the server to perform account assignment. Then, users will preserve account reference information in their own home accounts. The pair of the public/private key of pairing-based broadcast encryption is made as the owner of each account. For instance, $user_i$ computes the public key information $P_1^{(a_i)}, P_2^{(a_i)}, \cdots, P_n^{(a_i)}$ and private key. The resultant public key information is preserved in account $a_i$, and the private key of $user_j$ is encrypted with key $pub_j$, which is open to the public as a past public key cryptosystem; it is preserved in the home account of $user_j$.

Figure 5 shows the situation in which the data are preserved in account $a_i$. Public information for made broadcast encryption and information $a_i, a_j, \cdots, a_k$ of reference to account of $user_i$ who is owner is preserved. Moreover, the data naturally allocated in $a_i$ are preserved.

Because $user_i$ generates private key in each user's account $a_i$, $user_i$ has all users' private keys. However, from the condition; *A owner of each account has access authority for all data preserved in the account*, even owner of $a_i$ has other users' private keys, it is unquestionable because $user_i$ has at first the access authorities of all data in account $a_i$. Moreover, $user_i$ manages the private key where the user does not exist now because it produces many keys. The key is encrypted and transmitted to a new user's account, as described above, when the number of users increases.

### 5.2. Encrypting Data Entry

The data are encrypted and preserved in the server as follows when $user_i$ adds a new data entry $e$ which is shared among user set $S$. First, preservation account $a$ of $e$ is determined from the reference list of the user included in $S$




(a) 100 users (b) 1, 000 users (c) 10, 000 users

**Figure 4. Distribution of users**

according to the method described in section 3.2.2. Next, public keys of users in $S$ are gathered and $e$ is encrypted by broadcast encryption with there keys. Finally, the account reference list is updated for the user who was not referring to account $a$ to her/his referring in further.

## 6. Conclusion

In this paper, we proposed a novel mechanism using broadcast encryption and account assignment methods to reduce the decryption cost of Web applications. The proposed technique is easily applied to Web applications. Additional function of existing Web application server is unnecessary. We described how to solve two issues: 1) Who creates the public/private key pair in broadcast encryption? 2) How are the keys delivered? Through simulation and evaluation, we demonstrated our approach can keep the cost of decryption low for any user or any group. Especially, unnecessary decryption of ungrantable data was reduced to 10% or less.

In our approach, data and the sharing groups are expected to increase. Therefore, cancellation of the sharing relation according to data deletion remains as a subject for future research. Users will be able to decrease the number of account references because the sharing relation is canceled. However, for decreasing account references, information related to how much data that each user has access authority to in each preserved account is needed. Achievement of synchronization and the transaction on the encrypted database is necessary to manage such data on the server. Efficient and secure transaction processing can be achieved by preparing an encrypted log file and using the method described herein.

## Acknowledgment

This research was supported in part by Kyoto University Global COE Program “Informatics Education and Research Center for Knowledge Circulating Society.”

## References

[1] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 10 1999.

[2] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. *Lecture Notes in Computer Science*, 3621:258–275, November 2005.

[3] E. Damiani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Key management for multi-user encrypted databases. In *StorageSS ’05: Proceedings of the 2005 ACM workshop on Storage security and survivability*, pages 74–83. ACM, 2005.

[4] G. I. Davida, D. L. Wells, and J. B. Kam. A database encryption system with subkeys. *ACM Trans. Database Syst.*, 6(2):312–328, 1981.

[5] R. Dutta, R. Barua, and P. Sarkar. Pairing-based cryptographic protocols: A survey, 2004.

[6] E. Damiani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Selective data encryption in outsourced dynamic environments. *Electron. Notes Theor. Comput. Sci.*, 168:127–142, 2007.

[7] H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra. Executing sql over encrypted data in the database-service-provider model. In *SIGMOD ’02: Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 216–227, New York, NY, USA, 2002. ACM.

[8] B. Hore, S. Mehrotra, and G. Tsudik. A privacy-preserving index for range queries. In *VLDB’2004: Proceedings of the Thirtieth international conference on Very large data bases*, pages 720–730, 2004.

[9] X. Zou, Y. Karandikar, and E. Bertino. A dynamic key management solution to access hierarchy. *Int. J. Netw. Manag.*, 17(6):437–450, 2007.